WHAT IS CLAIMED IS:

1. A cache flush system, comprising:

a valid array unit configured to provide cache block information for an update algorithm and index information for a cache flush algorithm of at least one cache block in a prescribed state;

a storage unit configured to store tags and provide match address information for the update algorithm and tag information for the cache flush algorithm;

a bus snooper configured to perform the update algorithm for the tag storage unit and the valid array unit by monitoring a processor bus and by tracing a state of each cache memory; and

a cache flush unit configured to detect a system event, to perform the cache flush algorithm for corresponding cache blocks in the prescribed state.

2. The system of claim 1, wherein the storage unit comprises:

a tag RAM unit configured to store a plurality of tags for a prescribed index; and

a match logic unit coupled to the tag RAM unit configured to provide whether processor bus address matches a tag of corresponding address index by outputting the match address information, and wherein the match logic unit is configured to provide a tag that will make up address for a read transaction by outputting the tag information.

3. The system of claim 1, wherein the bus snooper performs the update algorithm by operating placement algorithm using cache block information from a prescribed logic operation.

4. The system of claim 3, wherein the prescribed operation is a divide and conquer AND tree (DCAND), wherein the placement algorithm traces a lower direction cache block under a condition that each output of the DCAND is '0' illustrating that empty cache block exists in a corresponding branch.

5. The system of claim 1, wherein the cache flush unit comprises:

an event detector configured to detect whether the system event occurs;

a cache flusher configured to generate a read transaction for a corresponding cache block according to the index information and the tag information to which a corresponding address is mapped, by performing the cache flush algorithm; and

a cache bus master configured to perform the cache flush algorithm for the cache memory by transferring the read transaction to each processor through outputting the generated read transaction to the processor bus.

6. The system of claim 5, wherein the cache flusher extracts a corresponding index by performing an address arrangement using index information from a prescribed logic operation, obtains the tag information from a match logic unit of the storage unit providing the index and maps the address by incorporating the index and the tag.

7. The system of claim 6, wherein the prescribed logic operation is a divide and conquer OR tree (DCOR), wherein the address arrangement traces a lower direction index under a condition that each output of the DCOR is '1' illustrating that occupied cache block exists in a corresponding branch.

8. The system of claim 1, wherein the prescribed state is a valid state including a modified state and an exclusive state, and wherein the cache flush unit generates a read transaction to perform the cache flush algorithm for the cache block in the valid state and outputs the read transaction to the processor bus.

9. A cache flush method, comprising:

updating status information by monitoring a transaction of a processor bus and tracing states of cache memory corresponding to each processor; and

flushing at least one cache block in a prescribed state among the cache blocks by detecting a prescribed event, generating a read transaction using the status information and outputting the generated read transaction.

10. The method of claim 9, wherein said updating status information comprises:

determining whether an attribute of the transaction is read by monitoring start of the transaction on the processor bus and by extracting an attribute of the transaction;

determining whether a share attribute is asserted from a processor except for a transaction master processor for the read attribute of the transaction, and wherein when the share attribute is asserted from the processor when a processor whose address matches the share assertion exists, clearing a corresponding valid bit of the address matched processor to set an invalid state and not setting a corresponding valid bit of the transaction master processor as a valid state.

11. The method of claim 10, wherein said updating status information comprises:

receiving cache block information using a placement process in a case where the share attribute is not asserted; and

storing tags in a location, designated by an index corresponding to the address in a certain cache block, according to the cache block information and setting a valid bit of the corresponding cache block as the valid state.

12. The method of claim 10, wherein when the attribute is not the read transaction said updating status information comprises:

determining whether the attribute is read-with-intent-to-modify;

determining whether a processor whose address matches exists and clearing the valid bit of the matched address processor as an invalid and not clearing the valid bit in a case where the processor whose address matches does not exist where the attribute of the transaction is read-with-intent-to-modify;

receiving cache block information using a placement process; and

storing a tag in a location designated by an index corresponding to the address in a certain cache block and setting a valid bit of a corresponding cache block as the valid state, according to the cache block information.

13. The method of claim 12, wherein when the attribute is not send-with-intent-to-modify said updating status information comprises:

determining whether the attribute is kill transaction;

determining whether a processor whose address matches exists, and clearing a valid bit of the address matched processor as an invalid state and not clearing the valid bit in a case where the processor whose address matches does not exist where the attribute is kill transaction;

receiving cache block information; and

storing a tag in a location, designated by an index corresponding to the address in a certain cache block and setting a valid bit of a corresponding cache block as a valid state, according to the cache block information.

14. The method of claim 13, wherein when the attribute is not the kill transaction said updating status information comprises:

determining whether snoop push is performed by being cast out through replacement operation of cache controller; and

clearing a corresponding valid bit for the address matched processor where the snoop push is performed by being cast out.

15. The method of claim 9, wherein said flushing comprises:

determining whether a valid array state for all processors is a valid state by receiving notification about whether the prescribed event occurs;

in a case where the valid array state for a first processor is the valid state as a result of determining whether the valid array state for the first processor is the valid state, determining whether a valid array state for a first cache block of the first processor is the valid state;

in a case where the valid array state for the first cache block of the first processor is the valid state, setting a value extracted by address arrangement as an index, receiving tag information corresponding to the index and extracting an address of the cache block;

performing cache flush for the cache block by generating a read transaction for the cache block corresponding to the extracted cache block address and by outputting the generated read transaction to a processor bus; and

ending the cache flush in a case where the cache block for which the cache flush is performed is a final cache block among cache blocks corresponding to a processor that is an object of the 'read' transaction and the processor is a final processor among said all processors.

16. The method of claim 15, wherein said performing cache flush comprises, in a case where the valid array state for the processor is not the valid state, determining whether a valid array state for a next processor is the valid state.

17. The method of claim 15, wherein said performing cache flush comprises, in a case where the processor is not the final processor, determining whether a valid array state for a next processor is the valid state.

18. The method of claim 15, wherein said performing cache flush comprises, in a case where the valid array state of the cache block is not a valid state, determining whether the valid array state for a next cache block is the valid state.

19. The method of claim 15, wherein said performing cache flush comprises, in a case where the cache block is not the final cache block, determining whether a valid array state for a next cache block is the valid state.

20. The method of claim 9, wherein the prescribed state is a valid state, and wherein the status information includes tag information and a valid status information.

21. A multi processor system, comprising:

a plurality of processors coupled to a processor bus;

at least one cache memory coupled to each processor;

a memory controller coupled to the processor bus; and

a cache flush system coupled to the processor bus, wherein the cache flush system comprises,

a first unit configured to provide cache status information;

a second unit coupled to the first unit and configured to update the cache status information; and

a third unit coupled to the first unit and configured to detect system events to perform cache flushing for corresponding cache blocks in a prescribed state responsive to the detected event.

22. The system of claim 21, wherein the second unit is configured to update the cache status information by monitoring the processor bus and by tracing a state of each cache memory.

23. The system of claim 21, wherein the first unit comprises:

a valid array unit configured to provide cache block information for an update algorithm and index information for a cache flush algorithm of at least one cache block in a prescribed state;

a tag storage unit configured to store tags and provide match address information for the update algorithm and tag information for the cache flush algorithm.